

# Distributed Scheduling for Real-Time Convergecast in Wireless Sensor Networks

XiaoHua Xu\*, Xiang-Yang Li\*<sup>†</sup>, ShaoJie Tang\*, XuFei Mao<sup>†</sup>

\*Department of Computer Science, Illinois Institute of Technology, Chicago, IL

<sup>†</sup>School of Software, Tsinghua University, Beijing, P.R.China

**Abstract**—Wireless sensor networks (WSNs) need to support real time periodic or sporadic queries of physical environments. In this work, we focus on the periodic queries. For each periodic query issued by control applications in a WSN, the data from the source sensors should be collected and/or aggregated to the control center within a certain end-to-end delay. We first propose almost-tight necessary conditions for a set of queries to be schedulable by a WSN. We then develop a family of efficient and effective data collection/aggregation algorithms that can meet the real-time requirement for quality of service (QoS) under resource constraints by addressing three tightly coupled tasks: (1) routing tree construction for data aggregation/collection, (2) link activity scheduling, and (3) packet scheduling at nodes. Our theoretical analysis for the schedulability of these algorithms show that they can achieve a constant fraction of the maximum schedulable load. For the case of overloaded networks where not all queries can be possibly satisfied, we propose an efficient algorithm for query selection that approximately maximizes the total weight of selected schedulable queries. Extensive simulations validating the proposed algorithms corroborate our theoretical analysis.



## 1 INTRODUCTION

Recent years have seen the emergence of wireless sensor networks (WSNs) where different types of sensors are deployed to monitor various aspects of the environment, such as monitoring light, temperature, acoustic and ammonia and so on. WSNs are also being deployed in a wide variety of other applications. For WSN applications, the data collected by the sensors are often streamed to a control center (called sink). A typical implementation will collect raw data from sensors and possibly perform in-network aggregation on data stream to relieve some communication burden (power and bandwidth) on the network. For most control applications, the semantics and the importance of data depend on the time when data are utilized. Thus, the observed events and consequently the data from the source sensors must be collected or aggregated at the control center within a certain delay. A key challenge then in WSNs is to meet the end-to-end delay requirement of control applications under wireless interferences and the severely limited resource constraints of WSNs.

A number of protocols have been proposed in the literature for data collection in WSNs that balance

the communication cost, delay, and reliability [16]. However, not much effort has been paid into the design of aggregation schemes that provide end-to-end performance guarantees for periodic queries. In this paper, we concentrate on designing effective scheduling of activities of nodes to satisfy multiple heterogeneous queries. Given are a set of sensor nodes and a distinguished sink node. The sink node issues a set of periodic queries, each has a period, initial release time and relative deadline requirement for receiving the answer back. The sink node expects to receive the corresponding (possibility aggregated) data from all sensor nodes in time. Given any interference model (note that we are not restricting ourself to one specific interference model), the objective is to jointly design a routing tree for each query, and an interference-aware schedule of activities for all nodes (*i.e.*, when to transmit and what packets to transmit) such that the deadlines of queries are met.

The problem of sporadic query scheduling in the network for data aggregation/collection under various interference models has been extensively studied recently [2], [6], [8], [12], [15], [20]–[23], [34], [36], [39]. At the same time, numerous significant results exist for various scheduling re-

lated problems such as scheduling periodical jobs at a single processor [17], [18], [41] and packet-level scheduling for the Internet [32], [42]. Surprisingly, only a few approaches [7], [14], [33] have studied the “real-time” data aggregation/collection scheduling in multi-hop WSNs. Unfortunately, although performing reasonably well in simulations for WSNs, these methods do not provide a theoretical performance assurance. The major hurdle may be due to the absence of flow conservation when in-network data aggregation is performed.

Compared with prior work, our main contributions deal with the schedulability test and effective scheduling algorithms. In summary, our main contributions are as follows:

#### A Necessary Condition for Schedulable Queries

We propose a necessary condition for a set of queries to be schedulable: Theorem 2 summarizes a necessary condition for data aggregation queries and Theorem 6 summarizes a necessary condition for data collection queries under various interference models. To address this challenge, we propose several novel concepts such as *initial load of a node*, *initial load of a region*, and *relay load*.

#### A Sufficient Condition for Schedulable Queries

We design efficient algorithms for constructing a routing tree for each of queries, scheduling node activities for each wireless node, and packet scheduling. We theoretically prove that the schedulable queries by our methods achieve a load that is within a constant factor of the maximum schedulable load. Based on the proposed algorithms, in Theorem 3, we present a sufficient condition for schedulability of data aggregation queries and in Theorem 7, we present a sufficient condition for schedulability of data collection queries under various interference models in WSNs.

**Overloaded Network** When the load of all queries exceed the network capacity (*i.e.*, the WSN is overloaded with queries from control applications), we propose an efficient query-selection algorithm by carefully selecting a subset of queries such that the total weight of selected queries (that are schedulable by our algorithms) is at least a constant fraction of the optimum solution.

**Simulation Results** We conduct extensive simulations to validate proposed algorithms. Our

simulation results in TinyOS corroborate our theoretical analysis.

The rest of the paper is organized as follows. Section 2 presents the system model. Section 4 and Section 5 respectively present schedulability results on data aggregation and collection queries under various interference models. Section 3 studies the query scheduling in overloaded networks. We present our simulation results in Section 6, review the related results in Section 7 and conclude the paper in Section 8.

## 2 SYSTEM MODELS

### 2.1 Network Model

Consider a WSN as a graph  $G = (V, E)$ , consisting of a set  $V$  of  $n$  sensor nodes where  $v_s \in V$  is the sink node, and  $E$  is the set of communication links. Two nodes can communicate with each other if they are within each others’ transmission range. To let two links transmit simultaneously, we must ensure they are interference free. In this work, we extensively study the schedulability of queries in a WSN under several commonly used interference models, such as Protocol Interference Model (PrIM), RTS/CTS Model, and Physical Interference Model (PhIM) or the Signal-to-Interference-plus-Noise Ratio model (SINR model). In PrIM [11], each node  $v_i$ , in addition to have a uniform transmission range (scaling to 1), has an *interference range*  $\rho$  such that any node  $v_j$  will be interfered by the signal from  $v_i$  if  $\|v_i - v_j\| \leq \rho$  and node  $v_j$  is *not* the intended receiver of the transmission from  $v_i$ . In the RTS/CTS model [1], for every pair of transmitter and receiver, all nodes that are within the interference range of either the transmitter or the receiver cannot transmit. In PhIM [9], there is a threshold value  $\beta > 0$ , such that a node  $v_j$  can correctly receive the data from a sender  $v_i$  if and only if the signal to interference plus noise ratio at the receiver satisfies

$$\text{SINR}(v_i, v_j) = \frac{P_i \cdot d_{i,j}^{\kappa}}{N_0 + \sum_{k \in I} P_k \cdot d_{k,j}^{\kappa}} \geq \beta.$$

Here  $d_{k,j}$  is the Euclidean distance  $\|v_k - v_j\|$ ,  $N_0 > 0$  is the background noise,  $P_i$  is the transmission power of node  $i$  (we assume the transmission power is a constant, *i.e.*,  $P_i = P$ ), while  $I$  is the set of actively transmitting nodes when node  $v_i$  is transmitting, and  $\kappa > 2$  is the path loss exponent.

## 2.2 Query Model

Assume the control application issues a set of heterogeneous queries, and all source nodes generate data reports periodically at specified data rates. In practice, queries could be different in many aspects. The  $i$ -th query can be characterized as follows: let  $\mathcal{S}_i \subseteq V$  denote a subset of source nodes each of which needs to answer this query. We assume that each source node  $v \in \mathcal{S}_i$  will generate a data unit to be collected to the sink  $v_s$  periodically. We assume that it takes  $\chi_i$  time to transmit a data unit for the  $i$ -th query over any link in the network. Here  $\chi_i$  could be different for different queries. For simplicity, we assume that  $\chi_i$  already takes into account the link reliability, data preparing time at nodes, and data size variety for answering queries.

The  $i$ -th query will be initially released at time  $\mathbf{a}_i$  and will have an end-to-end delay requirement  $\mathbf{d}_i$  for getting the answer. In other words, the sink should receive the answer before time  $\mathbf{f}_i = \mathbf{a}_i + \mathbf{d}_i$ . We assume that the  $i$ -th query has a period  $\mathbf{p}_i$ ; then, the  $t$ -th instance of this query will be released at time  $\mathbf{a}_i + (t-1) \cdot \mathbf{p}_i$  and the deadline for getting the answer is  $\mathbf{f}_i^t = \mathbf{a}_i + (t-1) \cdot \mathbf{p}_i + \mathbf{d}_i$ .

In this work, we focus on two types of queries: data aggregation and data collection. Data aggregation allows in-network fusion of data packets from different sensors enroute to the sink via multi-hop paths. For aggregating data, we assume that the data generated by a sensor node at a time  $t$  have a time-stamp  $t$ ; and we can only aggregate the data packets that have the same or similar time-stamps from a time period, depending on the control application. For simplicity, we assume that an intermediate node can aggregate multiple incoming packets into a single outgoing packet of the same size. The data collection query is to collect the raw data from every sensor node to the sink without any in-network processing.

Two different questions will be answered in this work. First, given a set of  $c$  queries  $\mathcal{Q}$  for data aggregation, each with its own period  $\mathbf{p}_i$ , processing time  $\chi_i$ , end-to-end deadline  $\mathbf{f}_i$ , and a set of sources nodes  $\mathcal{S}_i \subset V$ , whether the set of queries can be satisfied, and if so, design effective routing and scheduling algorithms to meet the specified requirements. The same question will be answered if we are given a set of queries for data collection instead. The second type of questions is to design routing

and scheduling protocols that will maximize the total weight of scheduled queries when we cannot schedule all queries successfully and each query is associated with a positive weight.

## 3 DROP OVERLOADED QUERIES

In this section, we study scheduling for an overloaded sensor network when not all arriving queries can be scheduled. Let us focus on the data collection queries: given a set of data collection queries  $\mathcal{Q}$ , assume the  $i$ -th query is associated with a weight  $\mathbf{w}_i$ . The objective is to select and schedule a subset of queries  $S \subseteq \mathcal{Q}$  to maximize the overall weight of the scheduled queries.

We reduce our problem to a *0-1 knapsack problem* as follows: given  $c$  items, the  $i$ -th query can be considered as an item of size  $\frac{|\mathcal{S}_i| \cdot \chi_i}{\mathbf{p}_i}$  and weight  $\mathbf{w}_i$ . The objective is to select a subset of items with total size at most  $C$  such that the weighted sum of all selected items is maximized. Here  $C$  is called the bag size. We will denote the 0-1 knapsack problem with bag size  $C$  by  $\text{KS}(C)$  for brevity. Then, our algorithm consists of two phases:

**Phase I:** we enumerate each single query whose load  $\frac{|\mathcal{S}_i| \cdot \chi_i}{\mathbf{p}_i}$  is no larger than 1 and select the one with the maximum weight as the first candidate solution;

**Phase II:** we use the solution for  $\text{KS}(\frac{0.69}{c_2(\mathcal{M}) \cdot c_4(\mathcal{M})})$  as the second candidate solution.

The final solution can be obtained by choosing the one with larger weight among these two candidate solutions. Please refer to Algorithm 1 for details. Note that we can design a joint routing and scheduling protocol to satisfy a set of data collection queries  $\mathcal{Q}$  under an interference model  $\mathcal{M}$ , if  $\sum_i \frac{|\mathcal{S}_i| \cdot \chi_i}{\mathbf{p}_i} \leq \frac{0.69}{c_2(\mathcal{M}) \cdot c_4(\mathcal{M})}$ . Therefore, it is easy to verify the correctness of our solution.

The challenge here is to derive an approximation bound on this solution. Recall that for any set of schedulable queries, we must have  $\sum_i \frac{|\mathcal{S}_i| \cdot \chi_i}{\mathbf{p}_i} \leq 1$ , which implies that the optimal solution for our problem is no larger than the optimal solution of  $\text{KS}(1)$ . Let  $\text{OPT}_{\text{KS}(1)}$  denote the optimal solution of  $\text{KS}(1)$ . The following lemma shows that the selected queries have weight at least a constant fraction of the weight of  $\text{OPT}_{\text{KS}(1)}$ .

**Lemma 1:** Let  $\mathbf{w}(A)$  denote the weight of the queries selected by Algorithm 1, and  $d = \frac{0.69}{c_2(\mathcal{M}) \cdot c_4(\mathcal{M})}$ , we have  $\frac{d}{2} \cdot \mathbf{w}(\text{OPT}_{\text{KS}(1)}) \leq \mathbf{w}(A)$ .

---

**Algorithm 1:** Maximum Weighted Query Selection
 

---

- 1:  $A_{[1]} := \{\arg \max_{i; \frac{|S_i| \cdot \chi_i}{p_i} \leq 1} \{\mathbf{w}_i\}\};$
  - 2:  $A_{[2]} :=$   
the solution returned by  $\text{KS}(\frac{0.69}{c_2(\mathcal{M}) \cdot c_4(\mathcal{M})});$
  - 3:  $A := \arg \max_{A \in \{A_{[1]}, A_{[2]}\}} \{\mathbf{w}(A)\};$
- 

Together with the fact that the optimum solution of our problem is no larger than  $\mathbf{w}(\text{OPT}_{\text{KS}(1)})$ , Theorem 1 immediately follows.

**Theorem 1:** Algorithm 1 is  $d/2$ -approximation for the maximum weighted query selection problem, where  $d = \frac{0.69}{c_2(\mathcal{M}) \cdot c_4(\mathcal{M})}$ .

In the previous discussions, we assumed that we will drop some queries when we cannot answer all queries in time. In practice, it may be possible to partially satisfy all queries, by carefully dropping some packets from some query flows (once every certain period), or dropping some packets from some data-source nodes. Dropping packets (temporally or spatially) is feasible for some applications because of the possible (temporal and/or spatial) correlation among data sensed by different sensors. Our algorithm can also be extended to deal with this case and details are omitted due to space limitations.

## 4 REAL-TIME SCHEDULE FOR AGGREGATIONS

This section deals with scheduling for data aggregation queries. We first propose both necessary conditions and sufficient conditions for schedulability of a given set of data aggregation queries. We then develop efficient routing protocols, link scheduling, and packet scheduling methods to satisfy a schedulable set of queries.

### 4.1 Necessary Conditions for Schedulability

Our study of necessary conditions and later sufficient conditions for schedulability rely on several novel concepts, namely *initial load* and *relay load* of a node (and/or a region). Let us first introduce the concept of *initial load*. Given a WSN  $G = (V, E)$  and a set of queries  $\mathcal{Q}$ , the initial load of a node  $u \in V$  is defined as  $\ell_{G, \mathcal{Q}}(u) = \sum_{u \in S_j} \frac{\chi_j}{p_j}$ , where  $\chi_j$  is the processing time,  $p_j$  is the period, and

$S_j \subseteq V$  is the set of source nodes in  $G$  for the  $j$ -th query. If we describe *region* as any continuous area in a two-dimensional plane, the initial load of a region  $g$  is defined as the summation of the initial loads of all nodes in this region  $g$ , i.e.  $\ell_{G, \mathcal{Q}}(g_{v,h}) = \sum_{u \in V(g)} \ell_{G, \mathcal{Q}}(u)$  where  $V(g)$  consists of all nodes from  $V$  lying in the region  $g$ .

In this section, we will focus on the initial load of a special region (called *interference-aware region*) which is a square in a two-dimensional plane, with the *interference-aware radius* as its side-length. Given an interference model  $\mathcal{M}$ , the interference-aware radius  $\lambda(\mathcal{M})$  is the maximum possible distance between two senders such that the corresponding two links will interfere with each other under  $\mathcal{M}$ . This means that two nodes can transmit concurrently without the interference of other links in the network if the distance between them is greater than  $\lambda(\mathcal{M})$ . We can compute  $\lambda(\mathcal{M})$  based on the parameters of the model  $\mathcal{M}$ . We then partition the two-dimensional plane by using a set of vertical lines  $a_i : x = i \cdot \lambda(\mathcal{M})$  where  $i \in \mathbb{Z}$  and horizontal lines  $b_j : y = j \cdot \lambda(\mathcal{M})$  where  $j \in \mathbb{Z}$ . Here  $\mathbb{Z}$  represents the set of all integers and  $i, j \in \mathbb{Z}$  is called the index of vertical line  $a_v$  and horizontal line  $b_h$ . Clearly, each square formed by a pair of neighboring vertical lines  $a_i, a_{i+1}$  and a pair of neighboring horizontal lines  $b_j, b_{j+1}$  is an *interference-aware region* denoted as  $g_{i,j}$ .

Observe that to schedule the nodes' transmissions, for a clique in which no two nodes can transmit concurrently, the summation of all nodes' initial loads in the clique can not exceed one. Generally, for any interference-aware region where the maximum number of nodes in that region that can transmit concurrently is a constant  $c_1(\mathcal{M})$ , hence the initial load of this region is at most  $c_1(\mathcal{M})$ .

On the other hand, for a data aggregation query (assume the  $j$ -th query), the sink node  $v_s \in V$  needs to receive at least  $\ell_j$  amount of data during every period  $p_j$ , which takes time  $\chi_j$  for aggregation. Obviously, for a set of queries  $\mathcal{Q}$ , the load at sink  $v_s$ , given by  $\sum_j \frac{\chi_j}{p_j}$ , is at most one if  $\mathcal{Q}$  can be answered in a delay specified by the queries and the network.

To sum up, we propose in Theorem 2 a necessary condition for a set of queries  $\mathcal{Q}$  to be *schedulable*, where a set of queries is schedulable *iff* they can be answered in time.

**Theorem 2:** If a set of data aggregation queries  $\mathcal{Q}$  is schedulable under an interference model  $\mathcal{M}$ ,



then the following conditions must be satisfied:

$$\begin{cases} \ell_{G,\mathcal{Q}}(g_{v,h}) & \leq c_1(\mathcal{M}), \forall g_{v,h} \\ \sum_j \frac{x_j}{p_j} & \leq 1 \end{cases} \quad (1)$$

Here  $\ell_{G,\mathcal{Q}}(g_{v,h})$  is the initial load of an interference-aware region  $g_{v,h}$ . Constant  $c_1(\mathcal{M}) \geq 1$  is the maximum number of nodes that can transmit concurrently in any interference-aware region under the interference model  $\mathcal{M}$ .

Henceforth all the proofs will be deferred to the appendix for your fluent reading of the main part before falling into the technique details.

Next, we derive the value of  $c_1(\mathcal{M})$  under various interference models. Note that for physical interference model, the interference-aware radius  $\lambda(\mathcal{M})$  is the same as the *maximum transmission radius*  $\mathbf{r} = \sqrt[n]{\frac{P}{N_0\beta}}$ . The maximum transmission radius  $\mathbf{r}$  can be perceived as a threshold for communication distances: a pair of nodes can possibly communicate and thus be connected *iff* their mutual distance is smaller than the threshold  $\mathbf{r}$ . In other words, a node  $u$  cannot transmit data to another node  $v$  which is more than  $\mathbf{r}$  distance away even in the absence of other concurrent transmissions.

**Lemma 2:** The constant  $c_1(\mathcal{M})$  in Theorem 2 is given as:

$$c_1(\mathcal{M}) = \begin{cases} \frac{16 \cdot \rho^2}{(\rho-1)^2} & \text{under PrIM} \\ 36 & \text{under RTS/CTS} \\ \lfloor \frac{2^\kappa \cdot P}{N_0 \beta^2} \rfloor & \text{under PhIM} \end{cases}$$

## 4.2 Efficient Algorithms for Scheduling Queries

Given a set of data aggregation queries subject to wireless interference constraints, we will design effective and efficient algorithms to answer it. Since our approach is based on the concept of connected dominating set (CDS) in a graph, let us define it first.

In a graph  $G = (V, E)$ , a subset  $V_0 \subseteq V$  is a *dominating set* (DS) if each node in  $V$  is either in  $V_0$  or adjacent to some node in  $V_0$ . Nodes in  $V_0$  are called dominators, whereas nodes not in  $V_0$  are called *dominatees*. A subset  $C \subseteq V$  is a *connected dominating set* (CDS), if  $C$  is a dominating set and  $C$  induces a connected subgraph in  $G$ .

### 4.2.1 General framework

Our framework for scheduling data aggregation queries under various interference models consists of several phases:

**Phase I:** For each query, construct a routing tree for data aggregation.

By using the routing tree  $\mathbf{T}_i$  for the  $i$ -th query, all data in the source nodes  $\mathcal{S}_i$  for this query can be routed to the sink  $v_s$ . Note that any routing tree  $\mathbf{T}_i$  is a special Steiner tree connecting the terminals of  $\mathcal{S}_i \cup \{v_s\}$  where node  $v_s \in V$  is the sink and  $\mathcal{S}_i$  is the set of source nodes. Since the routing tree will be used for data collection (Section 5.2) as well, we call it as the data gathering routing tree.

By using the data gathering routing tree, we will specify the data units to transmit in real time for each node. For the  $j$ -th query with a routing tree  $\mathbf{T}_i$ , during each period, first every leaf node in  $\mathbf{T}_i$  adds data to transmit, then every non-leaf node in  $\mathbf{T}_i$  generates *one* data unit and adds to its buffer upon receiving all data units from its children in  $\mathbf{T}_i$  for this period of the  $j$ -th query. Note that node  $u$  may receive data from multiple children, however, it only generates one unit of data by aggregating all received packets for the period with its own data, if any.

**Phase III:** Assign time for every node to transmit.

We will propose a *linear time assignment* scheme in which each node in a region is assigned with transmission time proportional to its *relay load*, defined as follows: Given a set of queries  $\mathcal{Q}$  and the corresponding data aggregation routing trees, we define the relay load of a node  $u$  as  $\mathcal{L}_{G,\mathcal{Q}}(u) = \sum_{\mathbf{T}_j \ni u} \frac{x_j}{p_j}$ . We then define the relay load of a region  $g$  as the summation of all nodes' relay loads in this region:  $\mathcal{L}_{G,\mathcal{Q}}(g_{v,h}) = \sum_{u \in V(g)} \mathcal{L}_{G,\mathcal{Q}}(u)$ , where  $V(g) \subseteq V$  is the set of all nodes from  $V$  lying in region  $g$ . The relay load contains both the initial load and the data load coming from routing data. Thus the relay load of a node can be perceived as the fraction of time for a node to be actively transmitting data using the routing structure. Therefore, a node with more packets (thus more relay load) will be assigned with more time to transmit.

**Phase IV:** Select data packets to transmit for each node.

We will use the *rate monotonic* method in this phase. Note that rate monotonic method is effective to ensure that every packet can catch its deadline for periodic jobs [18].

Let us now describe our methods for each phase separately.

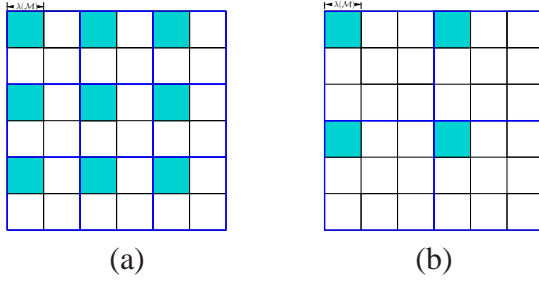


Fig. 1. Region Coloring: (a) for protocol interference model and RTS/CTS model; (b) for physical interference model.

#### 4.2.2 Constructing a data gathering routing tree

The constructions of routing trees are similar under various interference models. Given a communication graph  $G = (V, E)$ , we select a CDS  $\mathbf{T}_{CDS}$  of  $G$  by using an existing approach [26]. We then construct a *spanning tree*  $\mathbf{T}_G$  by connecting each node not in the CDS to a neighboring dominator.

For the  $i$ -th query, we prune every node  $u \in V$  and the corresponding link  $up(u)$  (the link from  $u$  to its parent  $p(u)$ ) in  $\mathbf{T}_G$  if the intersection between  $\mathcal{S}_i$  and the subtree of  $\mathbf{T}_G$  rooted at  $u$  (noted as  $\mathbf{T}_G^u$ ) is empty:  $\mathcal{S}_i \cap \mathbf{T}_G^u = \emptyset$ . The pruning operations result in a routing tree  $\mathbf{T}_i$  for the  $i$ -th query.

For PhIM, if the length of a communication link is very close to  $\mathbf{r}$ , then the SINR at the link's receiver is barely above the threshold, and the link's transmission will fail with high probability (*w.h.p.*). Therefore, a link whose length is very close to  $\mathbf{r}$  is not a good candidate for transmission in practice. Given a parameter  $\delta$ , if we connect every pair of nodes with distance at most  $\delta\mathbf{r}$  in the network, we can derive a reduced communication graph, denoted as  $G(V, \delta\mathbf{r})$ , a subgraph of the original communication graph  $G(V, \mathbf{r})$ . If  $G(V, \delta\mathbf{r})$  is connected, we can perform data transmissions in this subgraph. Therefore, under PhIM, we construct data gathering routing trees in a reduced communication graph  $G(V, \delta\mathbf{r})$  instead of a data gathering routing tree in  $G(V, \mathbf{r})$ .

After we construct a data gathering routing tree for each query, the second phase is to construct a real-time *transmission plan* for each node. We first need to determine which queries the node is involved with. Observe that a node  $u$  is involved in the  $j$ -th query if: (1)  $u$  is a source node for this query, *i.e.*,  $u \in \mathcal{S}_i$ , or (2)  $u$  is a relay node for this

query. In either case,  $u \in \mathbf{T}_i$ . Thus, we can test  $u \in \mathbf{T}_i$  to determine whether a node  $u$  is involved in the  $i$ -th query or not.

If a node  $u$  is involved in the  $j$ -th query, during each period  $\mathbf{p}_j$ , that node needs to add a packet for this query to its transmission plan. The added packet is either original packet or an aggregated one for the received data from all its children in the corresponding data gathering routing tree  $\mathbf{T}_j$ . For each node, we store the transmission plan to its buffer.

#### 4.2.3 Interference-aware node scheduling

After computing the transmission plans, each node may store some packets in its transmission plan. The third phase is to schedule (or assign) concrete time to each node for transmission, and to ensure interference-freeness at the same time. The proposed scheduling consists of two steps: (1) determine which region to select nodes from, called an *active region*; (2) determine which node from an active region to transmit.

First, we color all interference-aware regions such that any pair of neighboring regions with the same color are separated by  $K(\mathcal{M}) - 1$  regions, where  $K(\mathcal{M})$  is a constant depending on the interference model. Clearly, the chromatic number for this coloring method is  $c_2(\mathcal{M}) = K(\mathcal{M})^2$ . Specifically,  $c_2(\mathcal{M}) = 4$  under the protocol interference model, the RTS/CTS model, and  $c_2(\mathcal{M})$  is some other constant in the physical interference model (see Fig. 1). With the help of region coloring, we ensure that if only one node is selected from each of the interference-aware regions with the same color to transmit, we can ensure interference-freeness under the given interference model, irrespective of the positions of the receivers. Using this property, each time we let all regions with the same color be active to ensure interference-freeness.

Second, we assign transmission time to nodes in an active region. Clearly, a node with more relay load needs to be assigned with more time. We propose a linear time assignment scheme in which each node in an active region is assigned with transmission time proportional to its relay load. Given a time duration  $T$  (here  $T > \mathbf{p}_j, \forall j$ ) such that an interference-aware region  $g$  is active, we need to assign each node  $u$  in region  $g$  with transmission time  $T \cdot \frac{\mathcal{L}_{G, \mathcal{Q}}(u)}{\mathcal{L}_{G, \mathcal{Q}}(g_{v,h})}$ .

The details are shown in Algorithm 2 which is performed for every  $c_2(\mathcal{M}) \cdot T$  time duration. Then each region is active for exactly  $T$  time duration. When a region is active, we apply linear time assignment to each node in this region. Note that, the assignment does not require global coordination between different regions. Thus it can be implemented in a distributed manner efficiently.

---

**Algorithm 2:** Interference-aware node scheduling

---

**Input** : Routing trees for all queries

```

1  $K(\mathcal{M}) \leftarrow \lceil \sqrt{c_2(\mathcal{M})} \rceil$ ;
2 for each interference-aware region  $g_{v,h}$  where
    $v, h \in \mathbb{Z}$  and  $g_{v,h}$  contains nodes do
3   Assign the region with color:
4    $(v \bmod K(\mathcal{M})) \cdot K(\mathcal{M}) + h$ 
    $\bmod K(\mathcal{M})$ ;
5 for  $i = 1, \dots, K(\mathcal{M})$  and  $j = 1, \dots, K(\mathcal{M})$ 
   do
6   for each region  $g_{v,h}$  of the  $i \cdot K(\mathcal{M}) + j$ -th
     color where  $v, h \in \mathbb{Z}$ , and  $g_{v,h}$  contains
     nodes do
7     for each node  $u$  in region  $g_{v,h}$  do
8       assign the node with transmission
       time:  $T \cdot \frac{\mathcal{L}_{G,Q}(u)}{\mathcal{L}_{G,Q}(g_{v,h})}$ ;
9 return a set of transmission time for each
   node.
```

---

#### 4.2.4 Packet scheduling

When it is a node's transmission time, our fourth phase is to select packet(s) from the node's transmission plan to transmit.

We use a *rate monotonic* [18], [29] method to select packets from the node's transmission plan:

- 1) All packets of current period have lower priorities than that of all previous periods.
- 2) The priorities of all packets of any queries are assigned on a rate-monotonic basis. In other words, a packet of current instance for a query with a shorter period has a higher priority over the packet of current instance for a query with a longer period (at absolute time  $t$ , a packet is at current instance if it is produced during a time period containing  $t$ ).

Similarly, a packet of previous instance for a query with a shorter period has a higher

priority over a packet of previous instance for a query with a longer period. Ties are broken by lexicographic order  $\langle \text{current/previous}, \mathbf{p}_i, \text{ID} \rangle$ .

- 3) All packets of previous instances for the same query are scheduled on the first-in-first-out basis.

As proved in [18], the rate monotonic method can achieve optimum performance for each packet to be transmitted before deadline, if each node has utilization (the utilization can be seen as the ratio of relay load to the fraction of time it is assigned to) of at most  $n \cdot (2^{1/n} - 1)$  where  $n$  is the number of queries the node is involved. Note that  $n \leq c$ . For large  $n$ , we obtain the utilization bound of 69% means that as long as each node has utilization of less than 69%, all packets can make their deadlines.

#### 4.3 Sufficient Conditions for Schedulability

In Section 4.2, we proposed a family of algorithms to schedule periodic data aggregation queries. We prove that our algorithms are *feasible*. Here an algorithm is feasible for a given set of queries *iff* by using the algorithm, we can ensure interference-freeness as well as answer all queries within the deadline.

**Lemma 3:** The proposed algorithms in Section 4.2 can answer a set of data aggregation queries without interferences, if

$$\begin{cases} \mathcal{L}_{G,Q}(g_{v,h}) \leq 0.69/c_2(\mathcal{M}), \forall g_{v,h} \\ \sum_j \frac{x_j}{p_j} \leq 0.69 \end{cases} \quad (2)$$

Where  $\mathcal{L}_{G,Q}(g_{v,h})$  is the relay load of an interference-aware region  $g_{v,h}$ , and  $c_2(\mathcal{M})$  is the chromatic number for region coloring such that if we only select one node from each of the interference-aware regions with the same color to transmit, we can ensure interference-free under the interference model  $\mathcal{M}$ .

**Lemma 4:** The proposed algorithms in Section 4.2 can answer all queries within the deadline, if  $\mathbf{d}_i \geq c_2(\mathcal{M}) \cdot T \cdot 2R : \forall i$ . Here  $\mathbf{d}_i$  is the delay requirement for the  $i$ -th query,  $c_2(\mathcal{M})$  is given in Lemma 5, and  $R$  is the radius of communication graph  $G$ .

Observe that smaller  $T$  means that we could satisfy more queries with tighter deadlines. On the other hand, observe that, in Algorithm 2, each node will be assigned time  $T \cdot \frac{\mathcal{L}_{G,Q}(u)}{\mathcal{L}_{G,Q}(g_{v,h})}$  to transmit packets in its transmission plan (or buffer).

This time should be at least sufficient to transmit one packet. In other words,  $T$  should be bounded from below by a value. When the time to transmit a packet is treated as 1 unit, we have  $T \geq \lceil \max_{v,h} \max_{u \in g_{v,h}} \frac{\mathcal{L}_{G,Q}(g_{v,h})}{\mathcal{L}_{G,Q}(u)} \rceil$ .

The feasibility verification (Lemma 3 and 4) implies a sufficient condition for schedulability of a given set of queries:

**Theorem 3:** Equation (2) is a sufficient condition for schedulability of a set of data aggregation queries.

Next, we derive  $c_2(\mathcal{M})$ 's value under various interference models.

**Lemma 5:** The value  $c_2(\mathcal{M})$  used in Lemma 3 is

$$c_2(\mathcal{M}) = \begin{cases} 4 & \text{under PRIM \& RTS/CTS} \\ O(1) & \text{under PhIM} \end{cases}$$

#### 4.4 Main Theorem

In this section, we study the difference between the necessary condition (Theorem 2) and sufficient condition (Theorem 3) for schedulability of a set of data aggregation queries. We address the questions for two cases: (1) queries on all nodes, *i.e.*,  $S_i = V$  for the  $i$ -th query, and (2) queries on a subset of nodes, *i.e.*,  $S_i \subset V$  for the  $i$ -th query.

##### 4.4.1 Queries on All Nodes

For queries on all nodes, every node needs to report a packet in every period of the  $j$ -th query. Then the initial load of each node is  $\sum_j \frac{\chi_j}{p_j}$ . On the other hand, for data aggregation, each node only needs to transmit one packet during a period of the  $j$ -th query. Thus the relay load of a node is at most  $\sum_j \frac{\chi_j}{p_j}$  no matter what routing tree is used. Since every node's initial load can not exceed its relay load, they are the same for each node. As a corollary, for each interference-aware region, the relay load is the same as the initial load. Using this property, we can easily prove Theorem 4 below.

**Theorem 4:** When all nodes have data, a constant approximation ratio of  $c_1(\mathcal{M})c_2(\mathcal{M})/0.69$  can be achieved for schedulability of a set of data aggregation queries  $\mathcal{Q}$  under various interference models  $\mathcal{M}$ .

##### 4.4.2 Queries on Subset of Nodes

For queries on a subset of nodes, different routing structures (a set of data gathering routing trees)

for the given set of queries will have vast impact on the relay load of a region, even if the region's initial load is fixed. In an extreme case, the relay load may be large while the initial load is zero (see Figure 2). However, we can compare the difference between the necessary condition (Theorem 2) and sufficient condition (Theorem 3) for schedulability from another perspective.

**Lemma 6:** Given an interference model  $\mathcal{M}$  and a set of data aggregation queries  $\mathcal{Q}$ , using our routing algorithms in Section 4.2, the following condition

$$\begin{cases} \ell_{G,Q}(g_{v,h}) & \leq 0.69 / (2 \cdot c_2(\mathcal{M})), \forall g_{v,h} \\ \sum_i \frac{\chi_i}{p_i} & \leq \frac{0.69}{2 \cdot c_2(\mathcal{M}) \cdot (c_4(\mathcal{M}) - 1)} \end{cases} \quad (3)$$

Here  $c_4(\mathcal{M})$  is the maximum size of CDS inside an interference-aware region plus one.

**Corollary 1:** Given a set of data aggregation queries  $\mathcal{Q}$  under an interference model  $\mathcal{M}$ , Equation (3) is a sufficient condition for schedulability.

**Theorem 5:** When a subset of nodes have data for each query, we can achieve an approximation ratio of  $\max\{2c_1(\mathcal{M})c_2(\mathcal{M})/0.69, \frac{2 \cdot c_2(\mathcal{M}) \cdot (c_4(\mathcal{M}) - 1)}{0.69}\}$  for schedulability of a set of data aggregation queries  $\mathcal{Q}$  under various interference models.

We derive the value of  $c_4(\mathcal{M})$  under various interference models.

**Lemma 7:** The constant

$$c_4(\mathcal{M}) = \begin{cases} 8 \cdot (\rho + 4)^2 & \text{under PRIM} \\ 200 & \text{under RTS/CTS} \\ 200 & \text{under PhIM} \end{cases}$$

We next show by example that the sufficient condition given in Corollary 1 is (approximately) tight. Indeed, the approximation ratio is independent of  $c_4(\mathcal{M})$ .

In Figure 2, node  $v_s \in V$  is the sink. There are  $\sqrt{k}$  vertical evenly spaced lines with distance

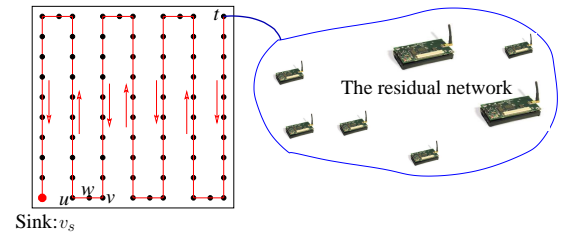


Fig. 2. An example for node placement in an interference-region.



$d_h = (1 + \epsilon)$  between consecutive lines (e.g., the distance between  $u$  and  $v$  is  $1 + \epsilon$ ). Additionally,  $\sqrt{k}-1$  nodes, like  $w$  between  $u$  and  $v$ , act as bridges to keep the network connectivity. Clearly, there are  $k = O(\rho^2)$  nodes deployed in the interference-aware region, and the size of CDS in this region is  $k-1$ , hence  $c_4(\mathcal{M}) = k$ . The residual network (all nodes outside of the region) is connected to sink  $v_s$  only through the red path from node  $t$  to  $v_s$ . We assume all sources nodes ( $\mathcal{S}_j$  for the  $j$ -th query) are located in the residual network.

To aggregate data to the sink  $v_s$ , we should strictly follow the red path as shown in Figure 2. It is easy to verify that the relay load of the interference-aware region is  $c_4(\mathcal{M}) \cdot \sum_i \frac{\chi_i}{\mathbf{p}_i}$  (the initial load is zero). Thus a necessary condition for schedulability for the example network is  $\sum_i \frac{\chi_i}{\mathbf{p}_i} \leq \frac{c_1(\mathcal{M})}{c_4(\mathcal{M})}$ . We can verify that the sufficient condition in Corollary 1 nearly match this necessary condition by a factor of at most  $c_1(\mathcal{M}) \cdot c_2(\mathcal{M})$  which is again independent of  $c_4(\mathcal{M})$ .

## 5 REAL-TIME SCHEDULE FOR COLLECTIONS

### 5.1 Necessary Conditions for Schedulability

Given a WSN with the communication graph  $G = (V, E)$  and a set of data collection queries  $\mathcal{Q}$ , assume the  $j$ -th query is associated with its time  $\chi_j$  needed by transmitting over a communication link, its period  $\mathbf{p}_j$ , its deadline  $d_j$ , and a set of source nodes  $\mathcal{S}_j \in \mathcal{S}$  in  $G$  that contains the initial raw data. For the  $i$ -th query, no matter what data collection routing tree is used, the sink node needs to receive all the raw data from  $\mathcal{S}_j$ . Thus, the initial load of sink node coming from the  $i$ -th query is exactly  $\frac{|\mathcal{S}_i| \cdot \chi_i}{\mathbf{p}_i}$ . If a set of queries  $\mathcal{Q}$  can be satisfied, the initial load of the sink node  $\sum_i \frac{|\mathcal{S}_i| \cdot \chi_i}{\mathbf{p}_i}$  is at most one. Therefore, we propose a necessary condition for schedulability as follows.

**Theorem 6:** If a set of data collection queries  $\mathcal{Q}$  under an interference model  $\mathcal{M}$  is schedulable, then

$$\sum_i \frac{|\mathcal{S}_i| \cdot \chi_i}{\mathbf{p}_i} \leq 1 \quad (4)$$

### 5.2 Efficient Algorithms for Scheduling Queries

In this section, we design effective algorithms for scheduling data collection queries under various

interference models. Data collection differs from aggregation in the sense that no in-network processing is allowed for data collection. Thus each node needs to transmit its raw data (if it has) and all received data towards the sink for data collection queries. The difference lies only in the transmission plan, thus we still apply the general framework proposed in Section 4.2.1 and only modify the phase for transmission plan construction.

**Phase I:** Construct a data processing routing tree in the network for each query (see the method in Section 4.2.2).

**Phase II:** Construct a transmission plan for each node.

For the  $j$ -th query, if a node  $u$  is a source node for this query, i.e.,  $u \in \mathcal{S}_j$ , during each period  $\mathbf{p}_i$ , node  $u$  needs add a packet of its raw data for this query to its transmission plan. In addition, if  $u$  is a non-leafing node in the data processing routing tree  $\mathbf{T}_j$ , during each period,  $u$  needs to route all packets it received from the subtree of  $\mathbf{T}_j$  rooted at  $u$  to its parent in  $\mathbf{T}_j$ . Thus, node  $u$  needs to add every packet it received to its transmission plan. Clearly, node  $u$  needs to add  $|D_T(u) \cap \mathcal{S}_j|$  packets during each period for the  $i$ -th query to its transmission plan.

**Phase III:** Assign transmission time to each node to ensure interference-freeness (see the method in Section 4.2.3).

**Phase IV:** Select data packets to transmit when it is a node's transmission time (see the method in Section 4.2.4).

### 5.3 Sufficient Conditions for Schedulability

In Section 5.2, we propose a family of algorithms to scheduling data collection queries. We next prove that our algorithms are feasible. Since we also use Algorithm 2 to assign transmission time to nodes, by Lemma 3, our algorithms can ensure interference-free under various interference models. We are only left to prove that our algorithms can answer all queries in time.

**Lemma 8:** The proposed algorithms in Section 5.2 can answer all data collection queries if

$$\sum_i \frac{|\mathcal{S}_i| \cdot \chi_i}{\mathbf{p}_i} \leq \frac{0.69}{c_2(\mathcal{M}) \cdot c_4(\mathcal{M})} \quad (5)$$

Here  $c_2(\mathcal{M})$  is given in Lemma 5, and  $c_4(\mathcal{M}) > 1$  is given in Lemma 7.

**Lemma 9:** The proposed algorithms in Section 5.2 can answer all queries within the deadlines, if  $\mathbf{d}_i \geq c_2(\mathcal{M}) \cdot T \cdot 2R$  for the  $i$ -th query where  $R$  is the radius of communication graph  $G$ .

Lemma 8 and 9 imply schedulability of the given set of queries. Thus, we propose a sufficient condition for schedulability.

**Theorem 7:** Equation (5) is a sufficient condition for schedulability of a set of data collection queries. We can also illustrate by an example (similar to Figure 2) that the sufficient condition in Theorem 7 is tight.

## 6 SIMULATION RESULTS

We randomly deploy a set of nodes  $\{v_1, \dots, v_n\}$  with transmission range 50 in an area of size  $400 \times 400$  (note that we always keep connectivity of the networks). For any pair of nodes  $v_i$  and  $v_j$ , there is a feasible link if  $|v_i v_j| \leq 50$ . In addition, each link  $(v_1, v_2)$  is associate with a quality variable  $q_{v_1 v_2}$ . Here, the value of  $q_{v_1 v_2}$ <sup>1</sup> is proportion to  $|v_i v_j|$ .

The sink node will generate up to 20 queries. A typical query message is  $\langle queryID, queryType_i, dataType_i, startingTime_i, timePeriod_i, dNodes \rangle$  where  $queryID$  is the unique query ID,  $queryType_i$  is the indicator distinguishing data collection and data aggregation,  $dataType_i$  is one of the data types,  $startingTime_i$  is the required starting time at which a node will start its duty circle,  $timePeriod_i$  is the duty circle period and  $dNodes$  contains IDs of all nodes who should respond to the  $i$ -th query. Table 1 shows a typical query in our simulation.

TABLE 1  
A typical query

$queryID$	$queryType_i$	$dataType_i$
1	1(data collection)	3
$startingTime_i$	$timePeriod_i$	$dNodes$
13:55:00 03/05/2010	5(mins)	5...29, 42...72, 81

The query with ID 1 shown in Table 1 requires every node with  $ID \in \{5 \dots 29, 42 \dots 72, 81\}$  to

start sampling the 3<sup>rd</sup> type data at 13 : 55 : 00 on 03/05/2010 with time period 5 minutes and send sample reading to the sink node under the collection operation.

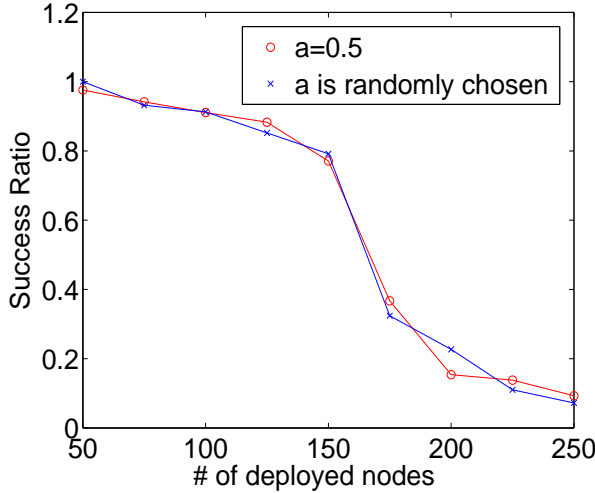
The main flow of our evaluation system is as follows: The sink node will generate a set of data aggregation (or collection) queries and broadcast it to the network one by one. The broadcast procedure will not stop until all nodes in the  $dNodes$  receive the  $i$ -th query correctly. Secondly, the sink node initiates to construct routing trees (based on the CDS) which cover all nodes (may need other nodes not in  $dNodes$  to relay) in  $dNodes$  will be constructed for data aggregation/collection. When starting time  $startingTime_i$  arrives, each node in  $nNodes$  will read the corresponding data ( $d_i$ ) repeatedly with time  $T_i$  and transmit via routing trees. The sink node will continue to analyze all received data packets for each period of each query. When all currently existing queries are satisfied, the sink node will release next query up to 20 queries totally. When, none of existing queries is satisfied, the algorithm will terminate.

We now evaluate the performance of our algorithms in different scenarios. In the first scenario, we vary the network size from 50 to 250 with step 25. For each query, we pick source nodes randomly or always choose a set of source nodes with half of the network size. Figure 3(a) shows the results when we either randomly pick the number of source nodes for each query or always randomly pick half of the nodes as source nodes. The success ratio is equal to the number of successful rounds divided the total rounds for existing queries.

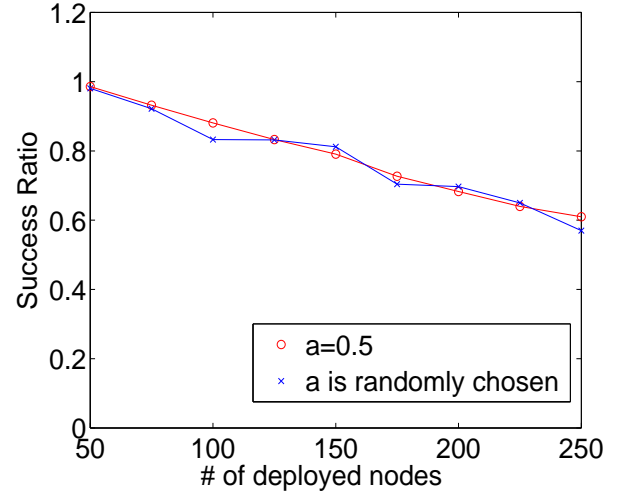
When the network size increases over 150, the success ratio will quickly drop from around 0.8 to 0.35. This is mainly caused by capacity bounds of CDS. The new packets from newly increased nodes (hence newly increased source nodes) lead CDS saturated such that many packets are dropped due to the buffer limit.

In the second scenario, we fix the network size and increase the number of source nodes in each query from 10 to 100 with step 10. The figure 3 (b) shows the success ratio when the network size is 100 and 200 respectively. As we can see, when the number of source node is small (less than 50), most of queries are satisfied. When the number of source node is larger than 50, the performance dropped quickly. In addition, there is no big difference

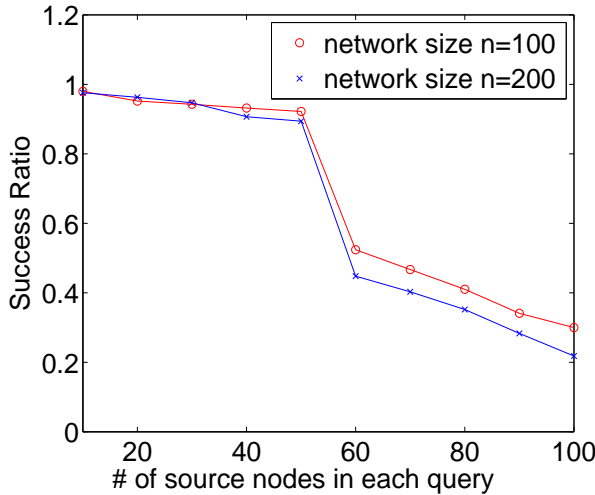
1. In Tossim [25], 0 dm denotes the best link quality and -115 dm is the critical value for a valid link.



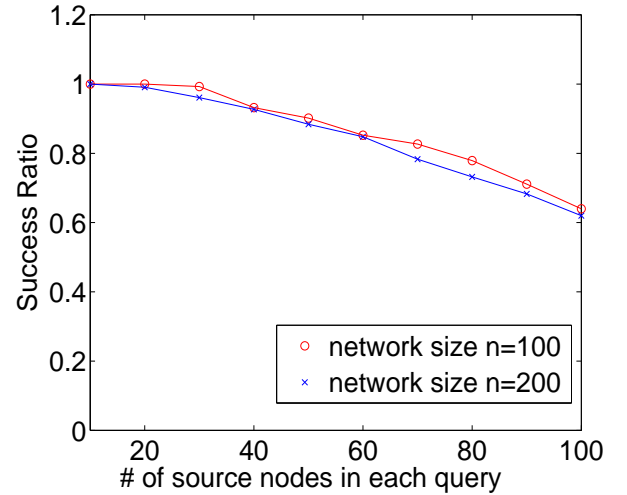
(a) Network size increases



(a) Network size increases



(b) # of sources increases



(b) # of sources increases

Fig. 3. The performance for data collection algorithm.  $a$  is the ratio of the number of chosen source nodes over the total number of nodes.

Fig. 4. The performance for data aggregation algorithm.  $a$  is the ratio of the number of chosen source nodes over the total number of nodes.

when network sizes (100 and 200 respectively) are different.

For data aggregation, we test our algorithm for data collection under the same network configuration. As we can see in Fig. 4 (a) and (b), the success ratio dropped smoothly when we either increase the network size or increase the number of source nodes while keeping the network size fixed. This illustrates that our algorithms work very well for data aggregation procedure, which is based on CDS.

## 7 RELATED WORK

### 7.1 Scheduling for Queries

Scheduling has been well studied in the literature for both single node and multi-node scenarios. For an

extensive survey on optimization and approximation results in deterministic sequencing and scheduling, readers may refer to [10]. In [18] was presented the first sufficient condition on whether a set of queries (or requests) can be answered by a rate monotonic method in a single processor. This was further extended in [17], [29]. Packet generalized processor sharing (P-GPS) [27], [28] provided flow control method in a more realistic packet transmission environment. Some statistical analysis of P-GPS was provided in [40]. A P-GPS server uses the packet

departing time as priority for scheduling packet transmissions. A number of packet scheduling methods (e.g., P-GPS [27], [28] and its variants such as weight fair queue (WFQ), Worst-case Fair WFQ [4]) have the desirable traffic fairness protection and can provide bounded delays. These scheduling methods could serve as a foundation for designing efficient and effective scheduling methods for multiple data collection/aggregation flows in multi-hop WSNs. In [19], the authors studied fair scheduling in wireless packet networks. However, we cannot directly apply the above methods because they neither considered the wireless interference constraints, nor the multicast or data aggregation flows. Additionally, they have potentially high computational complexity.

Recently, the authors in *et al.* [7] studied the real time query scheduling in WSNs by assuming a pre-given routing tree. Their results often suffer from the fact that different routing structures will have vast impact on the delay performances and flow data rates supported by a WSN. Given a routing structure, the ultimate goal at a node is to schedule the packet transmissions such that the end-to-end delay requirements are satisfied for all packets of all flows, whenever possible. The Earliest Deadline First (EDF) can provide optimal delay bounds at a single node. To provide the end-to-end delay of packets, several extensions of EDF were proposed in the literature, for example, EDF with traffic shaper [30]–[32] that can regulate the distorted traffic from the EDF scheduler to deal with the traffic burstiness. Unfortunately, using optimal traffic shaper is in general infeasible and will introduce additional packet delays. Another approach, such as deadline-curve based EDF (DC-EDF) [42], or similar one [5], is to judiciously adjust the local deadlines of packets at a node. Based on the traffic load and end-to-end deadlines, the DC-EDF scheme can guarantee end-to-end delay bounds and provide a schedulable region as large as that of RC-EDF [42].

## 7.2 Data Aggregation/Collection:

Data aggregation in WSN has been well studied recently [3], [13], [20]–[24], [36]. One sporadic query scheduling for data aggregation/collection with minimum delay has been proven to be NP-hard [6] and well studied in [12], [35], [39] [38].

A collision-free scheduling method for data collection is proposed in [16], aiming at optimizing

energy consumption and reliability.

## 8 CONCLUSIONS

We proposed joint design of a family of routing and packet scheduling schemes under different interference models. Most importantly, we theoretically proved that our algorithm can achieve constant approximation in terms of schedulability. We also studied the overloaded case where not all queries can be scheduled by proposing an efficient method for carefully selecting a subset of queries that maximizes the overall weight of the scheduled queries. In this case also, we theoretically proved that our proposed scheme can achieve constant approximation.

## REFERENCES

- [1] ALICHERY, M., BHATIA, R., AND LI, L. Joint channel assignment and routing for throughput optimization in multi-radio wireless mesh networks. In *ACM MobiCom*, (2005).
- [2] ANNAMALAI, V., GUPTA, S., AND SCHWIEBERT, L. On tree-based convergecasting in wireless sensor networks. In *IEEE WCNC*, (2003).
- [3] BEAVER, J., SHARAF, M., LABRINIDIS, A., AND CHRYSANTHIS, P. Location-Aware Routing for Data Aggregation in Sensor Networks. In *Geosensor Networks*, (2004).
- [4] BENNETT, J., AND ZHANG, H.  $WF^2Q$ : Worst-case fair weighted fair queueing. In *IEEE INFOCOM*, (1996).
- [5] BOURAS, C., AND SEVASTI, A. A delay-based analytical provisioning model for a QoS-enabled service. In *IEEE ICC*, (2006).
- [6] CHEN, X., HU, X., AND ZHU, J. Minimum Data Aggregation Time Problem in Wireless Sensor Networks. *MSN* (2005).
- [7] CHIPARA, O., LU, C., AND ROMAN, G. Real-time query scheduling for wireless sensor networks. In *IEEE RTSS*, (2007).
- [8] GANDHAM, S., ZHANG, Y., AND HUANG, Q. Distributed Minimal Time Convergecast Scheduling in Wireless Sensor Networks. In *IEEE ICDCS*, (2006).
- [9] GOUSSEVSKAIA, O. AND OSWALD, Y.A. AND WATTENHOFER, R. Complexity in geometric SINR. *ACM MobiHoc*, (2007).
- [10] GRAHAM, R., LAWLER, E., LENSTRA, J., AND KAN, A. Optimization and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics*, (1979).
- [11] GUPTA, P., AND KUMAR, P. The capacity of wireless networks. *IEEE Transactions on information theory*, (2000).
- [12] HUANG, S., WAN, P., VU, C., LI, Y., AND YAO, F. Nearly Constant Approximation for Data Aggregation Scheduling in Wireless Sensor Networks. In *IEEE INFOCOM*, (2007).
- [13] INTANAGONWIAT, C., ESTRIN, D., GOVINDAN, R., AND HEIDEMANN, J. Impact of Network Density on Data Aggregation in Wireless Sensor Networks. In *IEEE ICDCS*, (2002).
- [14] KALPAKIS, K., DASGUPTA, K., AND NAMJOSHI, P. Efficient algorithms for maximum lifetime data gathering and aggregation in wireless sensor networks. *Computer Networks*, (2003).
- [15] KESSELMAN, A., AND KOWALSKI, D. Fast distributed algorithm for convergecast in ad hoc geometric radio networks. *JPDC*, (2006).



- [16] LEE, H., AND KESHAVARZIAN, A. Towards Energy-Optimal and Reliable Data Collection via Collision-Free Scheduling in Wireless Sensor Networks. In *IEEE INFOCOM*, (2008).
- [17] LEUNG, J., AND WHITEHEAD, J. Complexity of fixed-priority scheduling of periodic, real-time tasks. *Performance Eval.* (1982).
- [18] LIU, C. L., AND LAYLAND, J. W. Scheduling algorithms for multiprogramming in a hard-real-time environment. *J. ACM*, (1973).
- [19] LU, S., BHARGHAVAN, V., AND SRIKANT, R. Fair scheduling in wireless packet networks. *IEEE/ACM Transactions on Networking*, (1999).
- [20] LUO, H., LIU, Y., AND DAS, S. K. Routing correlated data with fusion cost in wireless sensor networks. *IEEE TMC*, (2006).
- [21] LUO, H., LIU, Y., AND DAS, S. K. Distributed algorithm for en route aggregation decision in wireless sensor networks. *IEEE TMC*, (2009).
- [22] LUO, H., LUO, J., LIU, Y., AND DAS, S. K. Adaptive data fusion for energy efficient routing in wireless sensor networks. *IEEE Transactions on Computers*, (2006).
- [23] LUO, H., TAO, H., MA, H., AND DAS, S. K. Data fusion with desired reliability in wireless sensor networks. *IEEE TPDS*, (2010).
- [24] MADDEN, S., FRANKLIN, M., HELLERSTEIN, J., AND HONG, W. TAG: a Tiny AGgregation Service for Ad-Hoc Sensor Networks. *OSDI*, (2002).
- [25] N. L. Philip Levis. *TOSSIM: A Simulator for TinyOS Networks*, June 2003.
- [26] P.-J. WAN, K. A., AND FRIEDER, O. Distributed construction of connected dominating set in wireless ad hoc networks. In *IEEE INFOCOM*, (2002).
- [27] PAREKH, A., AND GALLAGER, R. A generalized processor sharing approach to flow control in integrated services networks: the single-node case. *IEEE/ACM Transactions on Networking*, (1993).
- [28] PAREKH, A., AND GALLAGHER, R. A generalized processor sharing approach to flow control in integrated services networks: the multiple node case. *IEEE/ACM Transactions on Networking*, (1994).
- [29] SHIH, W., LIU, J., AND LIU, C. Modified rate-monotonic algorithm for scheduling periodic jobs with deferred deadlines. *IEEE Transactions on Software Engineering*, (1993).
- [30] SIVARAMAN, V., AND CHIUSSI, F. Providing end-to-end statistical delay guarantees with earliest deadline first scheduling and per-hop traffic shaping. In *IEEE INFOCOM*, (2000).
- [31] SIVARAMAN, V., CHIUSSI, F., AND GERLA, M. Traffic shaping for end-to-end delay guarantees with EDF scheduling. In *IEEE IWQoS*, (2000).
- [32] SIVARAMAN, V., CHIUSSI, F., AND GERLA, M. End-to-end statistical delay service under GPS and EDF scheduling: A comparison study. In *IEEE INFOCOM*, (2001).
- [33] TAN, H., AND KÖRPEOĞLU, I. Power efficient data gathering and aggregation in wireless sensor networks. *ACM SIGMOD Record*, (2003).
- [34] UPADHYAYULA, S., ANNAMALAI, V., AND GUPTA, S. A low-latency and energy-efficient algorithm for convergecast in wireless sensor networks. In *IEEE GLOBECOM*, (2003).
- [35] WAN, P.-J., HUANG, S. C.-H., WANG, L., WAN, Z., AND JIA, X. Minimum-latency aggregation scheduling in multihop wireless networks. In *ACM MobiHoc*, (2009).
- [36] WANG, J., LIU, Y., AND DAS, S. K. Energy efficient data gathering in wireless sensor networks with asynchronous sampling. *ACM Transactions on Sensor Networks*, (2010).
- [37] WEGNER, G. Über endliche Kreispäckungen in der Ebene. *Studia Sci. Math. Hung.* (1986).
- [38] XU, X., WANG, S., MAO, X., TANG, S., AND LI, X. Y. A Delay Efficient Algorithm for Data Aggregation in Multi-hop Wireless Sensor Networks. In *IEEE TPDS*, (2010).
- [39] YU, B., LI, J., AND LI, Y. Distributed Data Aggregation Scheduling in Wireless Sensor Networks. In *IEEE INFOCOM*, (2009).
- [40] ZHANG, Z., TOWSLEY, D., AND KUROSE, J. Statistical analysis of generalized processor sharing scheduling discipline. In *ACM SigComm*, (1994).
- [41] ZHENG, Q., AND SHIN, K. On the ability of establishing real-time channels in point-to-pointpacket-switched networks. *IEEE Transactions on Communications*, (1994).
- [42] ZHU, K., ZHUANG, Y., AND VINIOTIS, Y. Achieving end-to-end delay bounds by EDF scheduling without traffic shaping. In *IEEE INFOCOM*, (2001).